Two-Way Finite Automata

Old and Recent Results

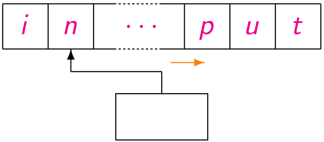
Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano

Automata and JAC 2012
La Marana, Corsica, France
September 19-21, 2012

UNIVERSITÀ DEGLI STUDI
DI MILANO

Finite State Automata



One-way version

At each step the input head is moved
one position to the right

- 1DFA: *deterministic* transitions
- 1NFA: *nondeterministic* transitions

A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a+b)^{n-1}a(a+b)^*$$

A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Check the nth symbol from the left!

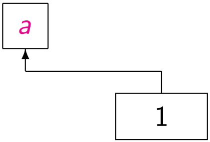
A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a+b)^{n-1}a(a+b)^*$$

Check the nth symbol from the left!

Ex. $n = 4$



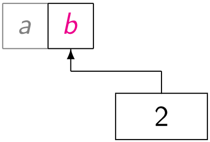
A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a+b)^{n-1}a(a+b)^*$$

Check the nth symbol from the left!

Ex. $n = 4$



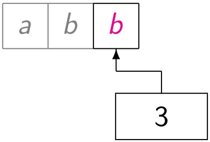
A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a+b)^{n-1}a(a+b)^*$$

Check the nth symbol from the left!

Ex. $n = 4$



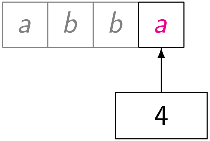
A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a+b)^{n-1}a(a+b)^*$$

Check the nth symbol from the left!

Ex. $n = 4$



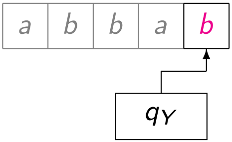
A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Check the nth symbol from the left!

Ex. $n = 4$



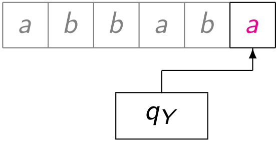
A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a+b)^{n-1}a(a+b)^*$$

Check the nth symbol from the left!

Ex. $n = 4$



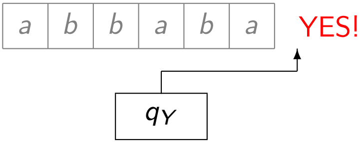
A Very Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$H_n = (a+b)^{n-1}a(a+b)^*$$

Check the nth symbol from the left!

Ex. $n = 4$



1DFA: $n + 2$ states

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the *n*th symbol from the right!

How to locate it?

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the *n*th symbol from the right!

How to locate it?

Use nondeterminism!

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the *n*th symbol from the right!

How to locate it?

Use nondeterminism!

Guess Reading the symbol *a* the automaton can guess that it is the *n*th symbol from the right

Verify In the next steps the automaton verifies such a guess

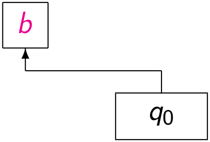
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



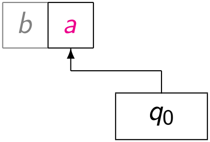
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



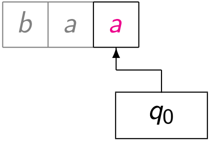
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



guess
4th symbol from the right

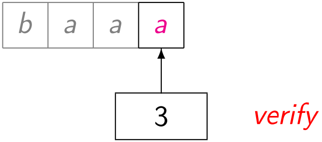
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



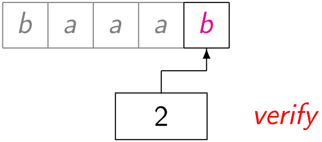
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



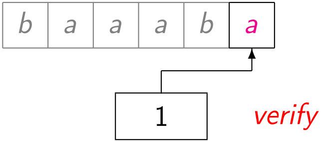
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



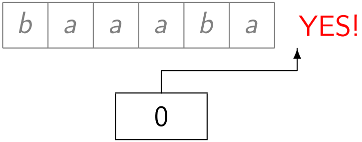
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$I_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



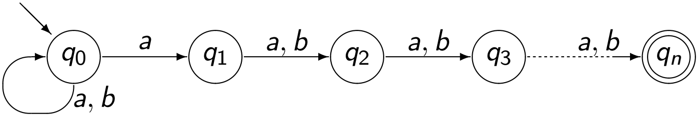
1NFA: $n + 1$ states

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a(a + b)^{n-1}$$

Check the nth symbol from the right!

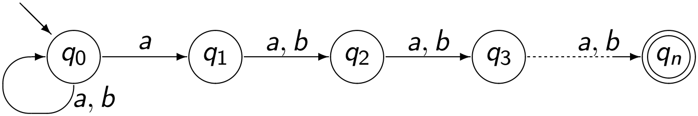


A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!



Very nice!

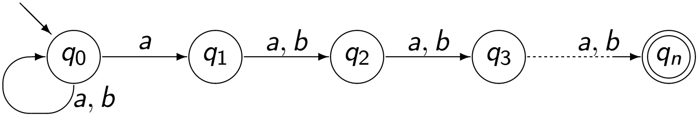
...but I need a *deterministic* automaton...

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!



Very nice!

...but I need a *deterministic* automaton...

Remember the previous n input symbols!

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$

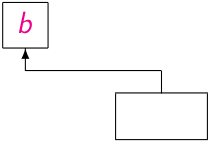
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



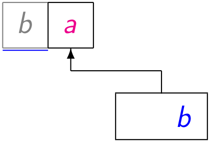
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



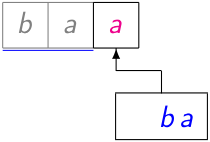
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the *n*th symbol from the right!

Ex. $n = 4$



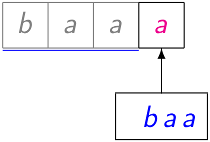
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



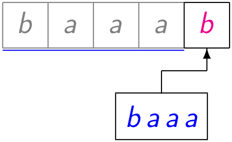
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



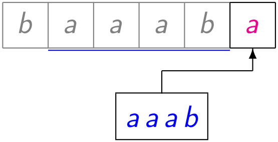
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



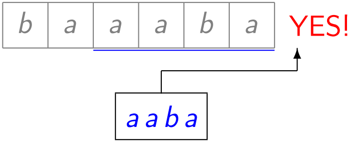
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the *n*th symbol from the right!

Ex. $n = 4$



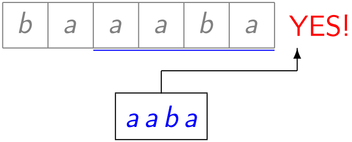
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



1DFA: 2^n states

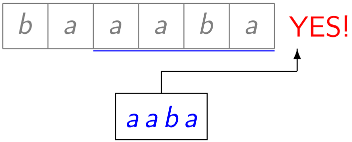
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$I_n = (a+b)^* a (a+b)^{n-1}$$

Check the *n*th symbol from the right!

Ex. $n = 4$



1DFA: 2^n states

...but I need a smaller deterministic automaton...

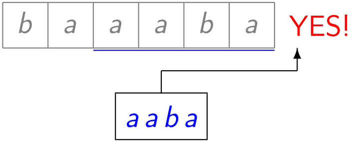
A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$I_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

Ex. $n = 4$



1DFA: 2^n states

...but I need a smaller deterministic automaton...

This is the smallest one!

However...

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the *n*th symbol from the right!

...if the head can be moved back...

A Preliminary Example

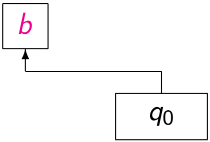
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

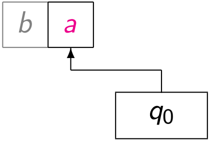
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

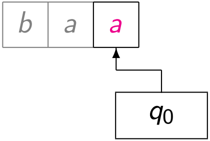
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a(a + b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

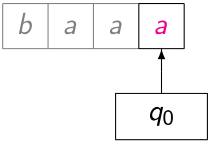
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

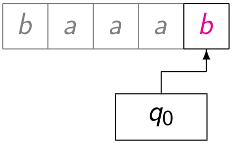
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

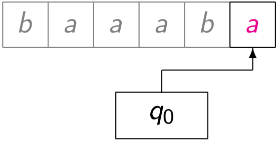
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a(a + b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

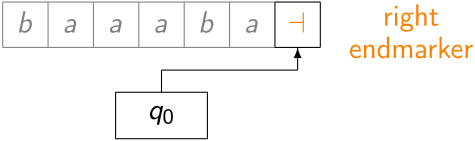
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$I_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

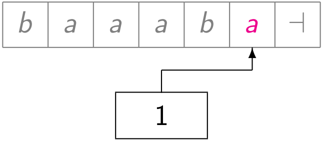
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

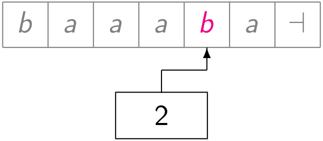
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a(a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

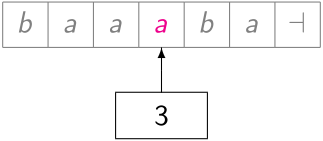
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



A Preliminary Example

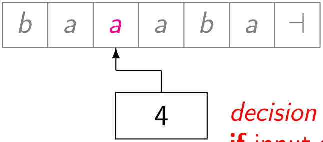
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



decision
if input symbol $= a$ **then accept**
else reject

A Preliminary Example

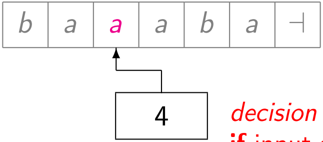
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



YES!

decision
if input symbol $= a$ **then accept**
else reject

A Preliminary Example

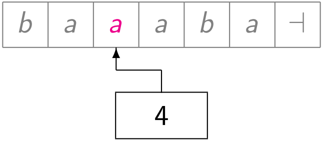
$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

Check the nth symbol from the right!

...if the head can be moved back...

Ex. $n = 4$



YES!

Two-way deterministic automaton (2DFA): $n + \dots$ states

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Check the nth symbol from the right!

Summing up, l_n is accepted by

- a 1NFA and a 2DFA with approximatively the same number of states $n + \ldots$
- each 1DFA is exponentially larger ($\geq 2^n$ states)

A Preliminary Example

$\Sigma = \{a, b\}$, fixed $n > 0$:

$$l_n = (a+b)^* a (a+b)^{n-1}$$

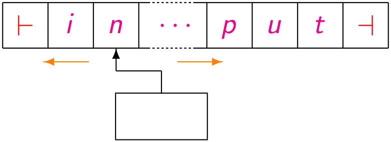
Check the *n*th symbol from the right!

Summing up, l_n is accepted by

- a 1NFA and a 2DFA with approximatively the same number of states $n+_{\ldots}$
- each 1DFA is exponentially larger ($\geq 2^n$ states)

In this example,
nondeterminism can be removed using two-way motion keeping approximatively the same number of states

Two-Way Automata: Technical Details



- Input surrounded by the *endmarkers* ⊢ and ⊣
- Moves
  - to the *left*
  - to the *right*
  - *stationary*
- Initial configuration
- Accepting configuration
- Infinite computations are possible
- *Deterministic* (2DFA) and *nondeterministic* (2NFA) versions

1DFA, 1NFA, 2DFA, 2NFA

What about the power of these models?

1DFA, 1NFA, 2DFA, 2NFA

What about the power of these models?

They share the same computational power, namely they characterize the class of *regular languages*,

1DFA, 1NFA, 2DFA, 2NFA

What about the power of these models?

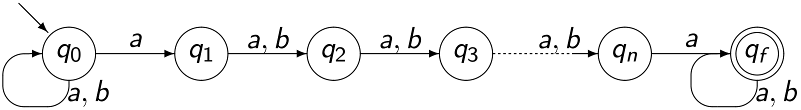
They share the same computational power, namely they characterize the class of *regular languages*, however...

...some of them are more succinct

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

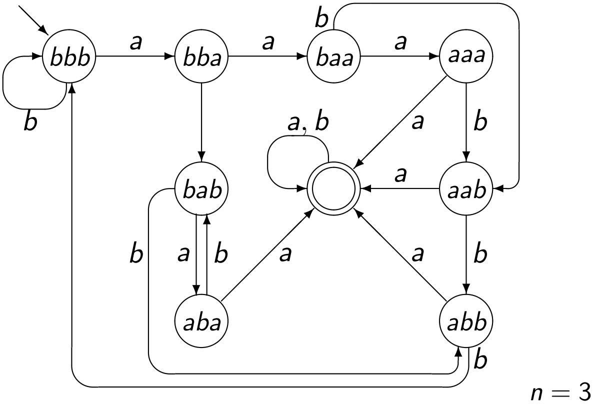
1NFA: $n + 2$ states

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



1NFA: $n+2$ states

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



Minimum 1DFA: $2^n + 1$ states

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

2DFA ?

Even scanning from the right it seems that
we need to remember a "window" of n symbols

We use a different technique!

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

2DFA ?

Even scanning from the right it seems that
we need to remember a "window" of n symbols

We use a different technique!

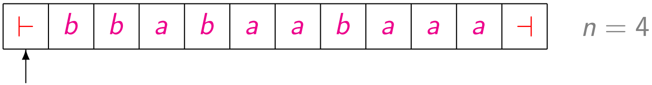
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

2DFA ?

Even scanning from the right it seems that
we need to remember a “window” of n symbols

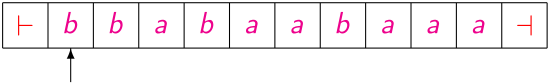
We use a different technique!

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



while input symbol $\neq a$ **do** move to the right

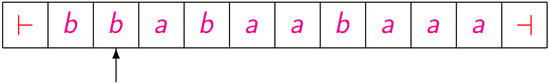
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right

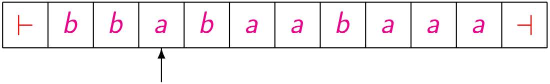
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

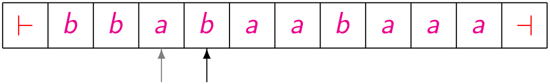


$n = 4$

while input symbol $\neq a$ **do** move to the right

move n squares to the right

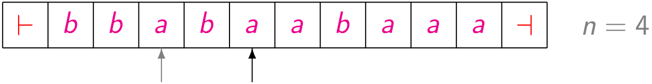
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right

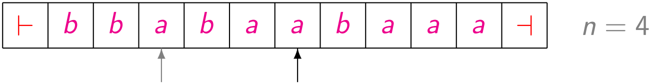
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



while input symbol $\neq a$ **do** move to the right

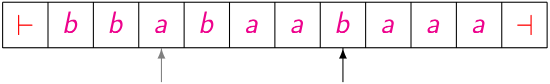
move n squares to the right

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



while input symbol $\neq a$ **do** move to the right
move n squares to the right

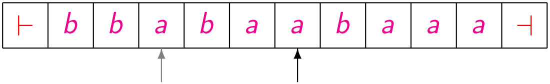
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n - 1$ cells to the left
repeat from the first step

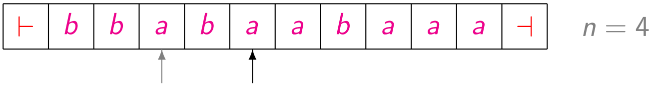
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

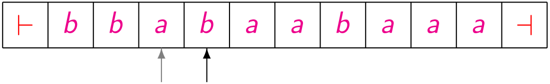
while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n-1$ cells to the left
repeat from the first step

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n-1$ cells to the left
repeat from the first step

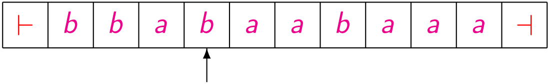
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n - 1$ cells to the left
repeat from the first step

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right

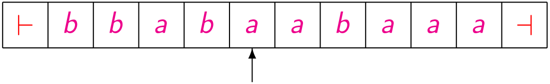
move n squares to the right

if input symbol $= a$ **then accept**

else move $n-1$ cells to the left

repeat from the first step

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right

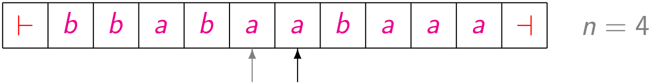
move n squares to the right

if input symbol $= a$ **then accept**

else move $n-1$ cells to the left

repeat from the first step

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



while input symbol $\neq a$ **do** move to the right

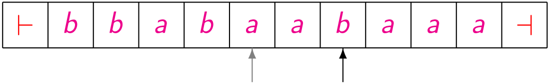
move n squares to the right

if input symbol $= a$ **then accept**

else move $n-1$ cells to the left

repeat from the first step

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right

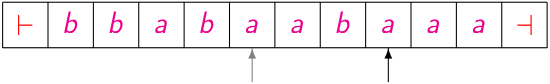
move n squares to the right

if input symbol $= a$ **then accept**

else move $n-1$ cells to the left

repeat from the first step

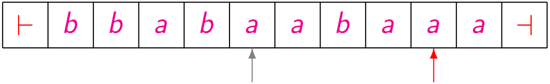
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n-1$ cells to the left
repeat from the first step

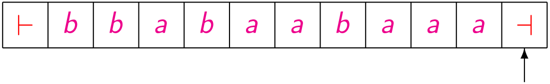
Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n-1$ cells to the left
repeat from the first step

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n-1$ cells to the left
repeat from the first step
Exception: **if** input symbol $= \dashv$ **then reject**

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

$\vdash$	b	b	a	b	a	a	b	a	a	a	$\dashv$

$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n-1$ cells to the left
repeat from the first step
Exception: **if** input symbol $= \dashv$ **then reject**

2DFA: $2n + \ldots$ states

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

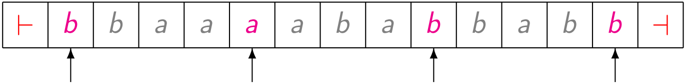
A different algorithm

$\vdash$	b	b	a	a	a	a	b	a	b	b	a	b	b	$\dashv$

$n = 4$

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

A different algorithm

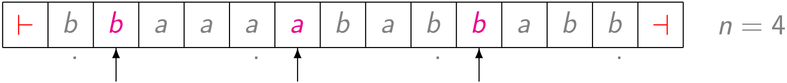


$n = 4$

Check positions k s.t. $k \equiv 1 \pmod{n}$

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

A different algorithm

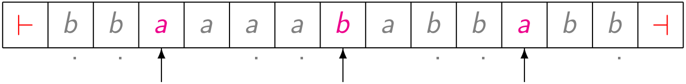


Check positions k s.t. $k \equiv 1 \pmod{n}$

Check positions k s.t. $k \equiv 2 \pmod{n}$

Main Example: $L_n = (a+b)^*a(a+b)^{n-1}a(a+b)^*$

A different algorithm



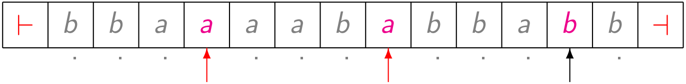
$n = 4$

Check positions k s.t. $k \equiv 1 \pmod{n}$
Check positions k s.t. $k \equiv 2 \pmod{n}$

...

Main Example: $L_n = (a+b)^*a(a+b)^{n-1}a(a+b)^*$

A different algorithm

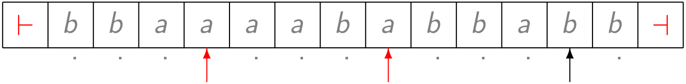


$n = 4$

Check positions k s.t. $k \equiv 1 \pmod n$
Check positions k s.t. $k \equiv 2 \pmod n$
...
Check positions k s.t. $k \equiv n \pmod n$

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

A different algorithm



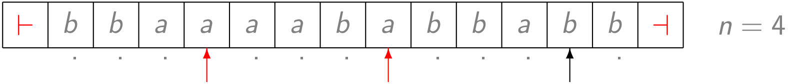
$n = 4$

Check positions k s.t. $k \equiv 1 \pmod{n}$
Check positions k s.t. $k \equiv 2 \pmod{n}$
...
Check positions k s.t. $k \equiv n \pmod{n}$

Even this strategy can be implemented using $O(n)$ states!

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

A different algorithm



Check positions k s.t. $k \equiv 1 \pmod{n}$
Check positions k s.t. $k \equiv 2 \pmod{n}$
...
Check positions k s.t. $k \equiv n \pmod{n}$

Even this strategy can be implemented using $O(n)$ states!

Sweeping automata:

- Deterministic transitions
- Head reversals *only at the endmarkers*

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Summing up,

- L_n is accepted by
  - a 1NFA
  - a 2DFA
  - a sweeping automaton

  with $O(n)$ states
- Each 1DFA is exponentially larger

Also for this example,
nondeterminism can be removed using two-way motion
keeping a linear number of states

Is it always possible
to replace nondeterminism by two-way motion
without increasing too much the size?

Main Example: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Summing up,

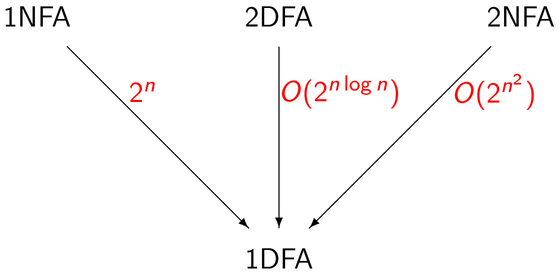
- L_n is accepted by
  - a 1NFA
  - a 2DFA
  - a sweeping automaton

  with $O(n)$ states
- Each 1DFA is exponentially larger

Also for this example,
nondeterminism can be removed using two-way motion
keeping a linear number of states

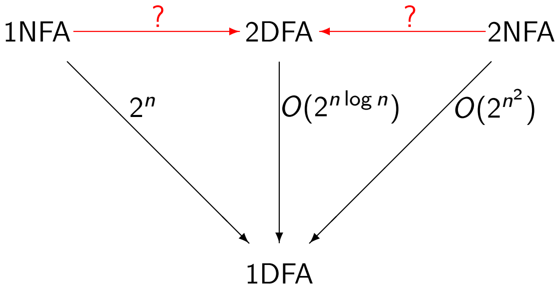
Is it always possible
to replace nondeterminism by two-way motion
without increasing too much the size?

Costs of the Optimal Simulations Between Automata



[Rabin&Scott '59, Shepardson '59, Meyer&Fischer '71, . . .]

Costs of the Optimal Simulations Between Automata

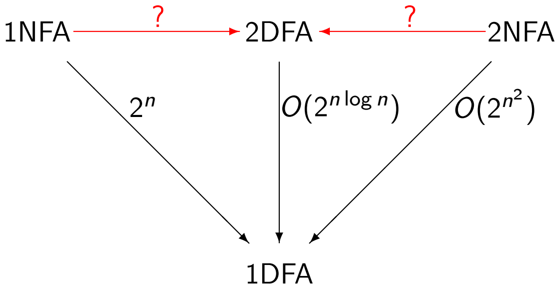


[Rabin&Scott '59, Shepardson '59, Meyer&Fischer '71, . . .]

Question

How much the possibility of moving the input head forth and back is useful to eliminate the nondeterminism?

Costs of the Optimal Simulations Between Automata

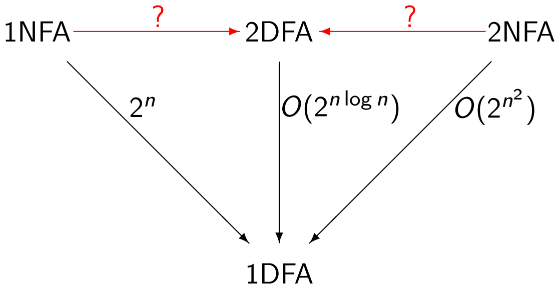


Problem ([Sakoda&Sipser '78])

Do there exist polynomial simulations of

- *1NFAs by 2DFAs*
- *2NFAs by 2DFAs ?*

Costs of the Optimal Simulations Between Automata



Problem ([Sakoda&Sipser '78])

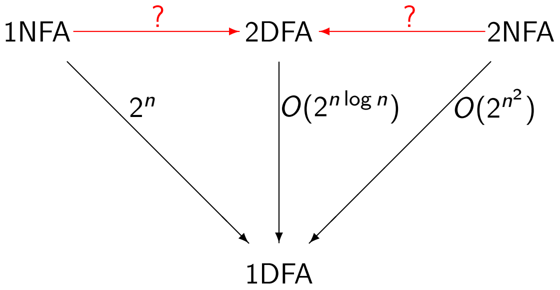
Do there exist polynomial simulations of

- *1NFAs by 2DFAs*
- *2NFAs by 2DFAs ?*

Conjecture

These simulations are not polynomial

Costs of the Optimal Simulations Between Automata



- Exponential upper bounds
  deriving from the simulations of 1NFAs and 2NFAs by 1DFAs
- Polynomial lower bound
  $\Omega(n^2)$ for the cost of the simulation of 1NFAs by 2DFAs
  [Chrobak '86]

Sakoda and Sipser Question

- Very difficult in its general form
- Not very encouraging obtained results:

  Lower and upper bounds too far
  (Polynomial vs exponential)

- Hence:

  Try to attack restricted versions of the problem!

NFAs vs 2DFAs: Restricted Versions

(i) Restrictions on the resulting machines (2DFAs)

- sweeping automata [Sipser '80]
- oblivious automata [Hromkovič&Schnitger '03]
- "few reversal" automata [Kapoutsis '11]

(ii) Restrictions on the languages

- unary regular languages [Geffert Mereghetti&P '03]

(iii) Restrictions on the starting machines (2NFAs)

- outer nondeterministic automata [Guillon Geffert&P '12]

NFAs vs 2DFAs: Restricted Versions

(i) Restrictions on the resulting machines (2DFAs)

- sweeping automata [Sipser '80]
- oblivious automata [Hromkovič&Schnitger '03]
- "few reversal" automata [Kapoutsis '11]

(ii) Restrictions on the languages

- unary regular languages [Geffert Mereghetti&P '03]

(iii) Restrictions on the starting machines (2NFAs)

- outer nondeterministic automata [Guillon Geffert&P '12]

NFAs vs 2DFAs: Restricted Versions

(i) Restrictions on the resulting machines (2DFAs)

- sweeping automata [Sipser '80]
- oblivious automata [Hromkovič&Schnitger '03]
- "few reversal" automata [Kapoutsis '11]

(ii) Restrictions on the languages

- unary regular languages [Geffert Mereghetti&P '03]

(iii) Restrictions on the starting machines (2NFAs)

- outer nondeterministic automata [Guillon Geffert&P '12]

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

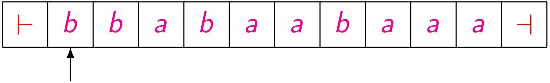
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$

$\vdash$	b	b	a	b	a	a	b	a	a	a	$\dashv$

$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

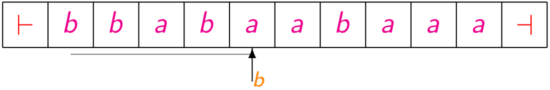
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

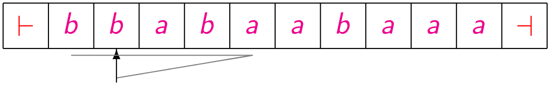
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

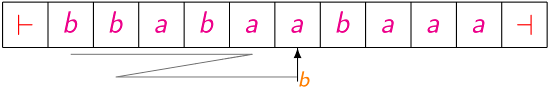
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

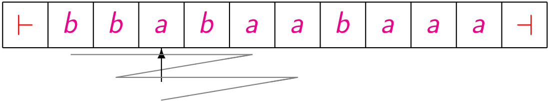
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

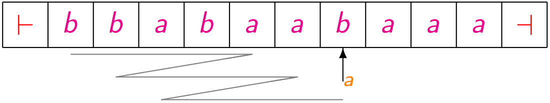
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

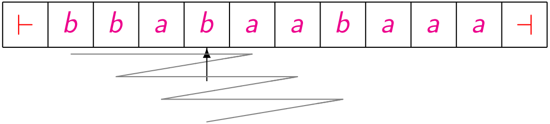
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

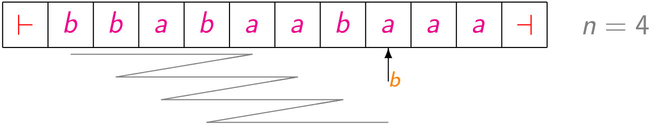
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

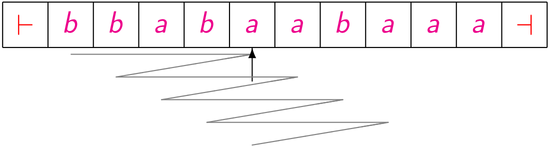
$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

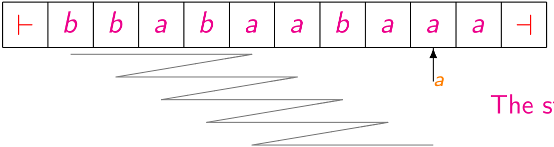
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^*a(a+b)^{n-1}a(a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$

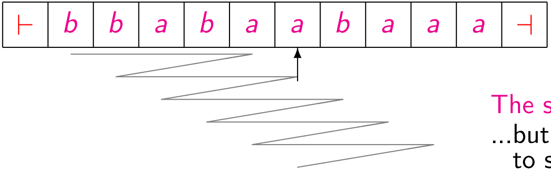


$n = 4$

The string can be accepted!

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



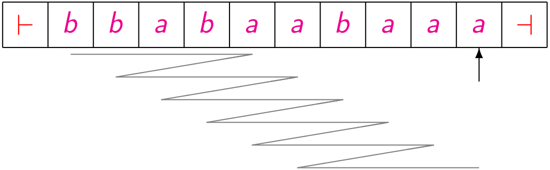
$n = 4$

The string can be accepted!

...but our automaton continues to scan

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

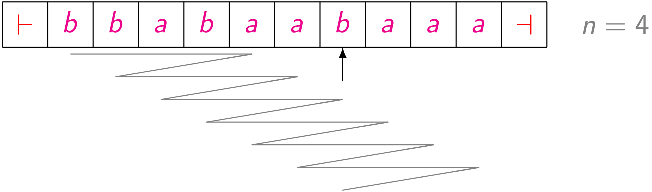
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

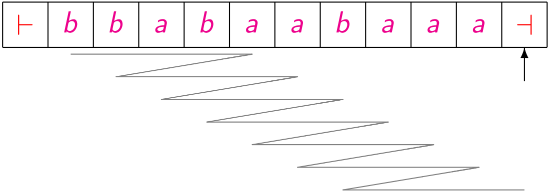
$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

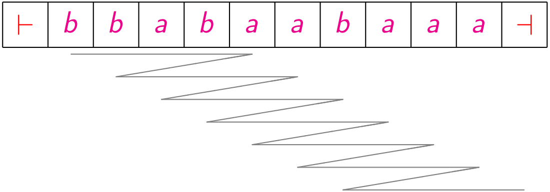
Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



$n = 4$

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$

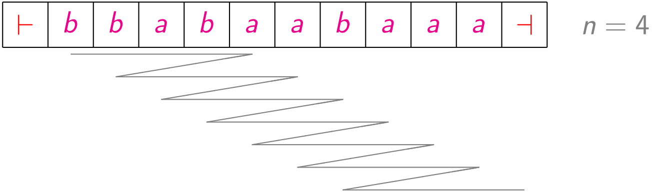


$n = 4$

Even in this case $O(n)$ states!

$L_n = (a+b)^*a(a+b)^{n-1}a(a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



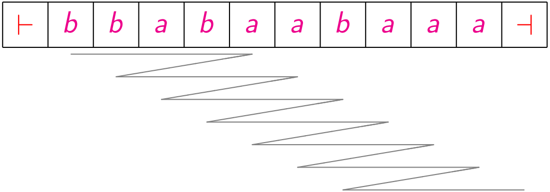
Even in this case $O(n)$ states!

Oblivious Automata:

- Deterministic transitions
- Same "trajectory" on all inputs of the same length

$L_n = (a+b)^*a(a+b)^{n-1}a(a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



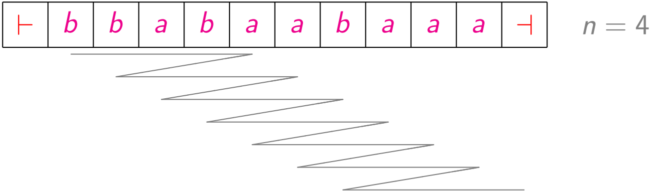
$n = 4$

Number of head reversals:
On input of length m:

- This technique uses about $2m$ reversals, a *linear number* in the input length
- The "sweeping" algorithm uses about $2n$ reversals, a *constant number* in the input length

$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$ Again!

Naïf algorithm: compare input positions i and $i+n$, $i = 1, 2, \ldots$



Number of head reversals:

On input of length m:

- This technique uses about $2m$ reversals, a *linear number* in the input length
- The "sweeping" algorithm uses about $2n$ reversals, a *constant number* in the input length

Another Restricted Model

"Few Reversal" Automata [Kapoutsis '11]:

- On input of length m the number of reversals is $o(m)$, i.e., sublinear
- We consider only the *deterministic case*

Theorem ([Kapoutsis&P '12])

Each 2DFA using $o(m)$ reversals actually uses $O(1)$ reversals

Another Restricted Model

"Few Reversal" Automata [Kapoutsis '11]:

- On input of length m the number of reversals is $o(m)$, i.e., sublinear
- We consider only the *deterministic case*

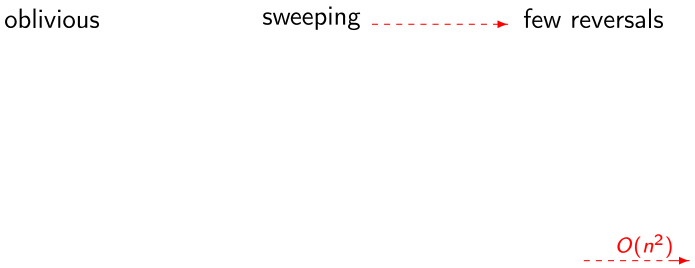
Theorem ([Kapoutsis&P '12])

Each 2DFA using $o(m)$ reversals actually uses $O(1)$ reversals

Restricted Models: Separations

oblivious sweeping few reversals

Restricted Models: Separations

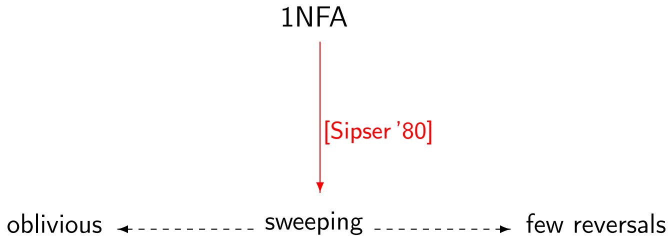


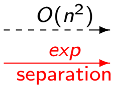
Restricted Models: Separations

oblivious ← sweeping → few reversals

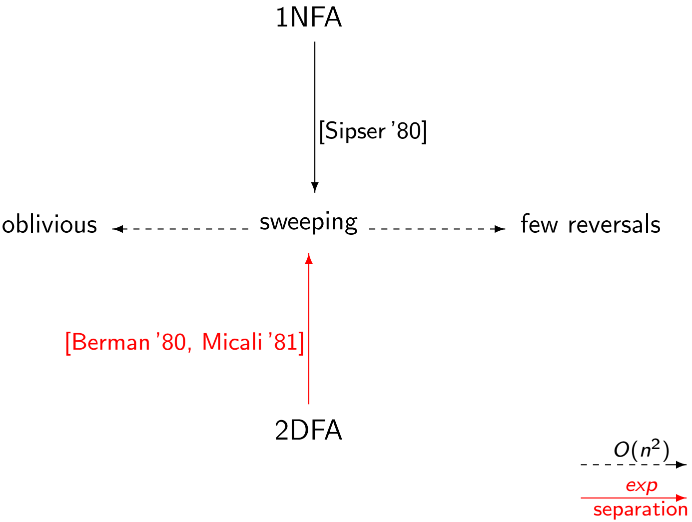
$O(n^2)$ →

Restricted Models: Separations

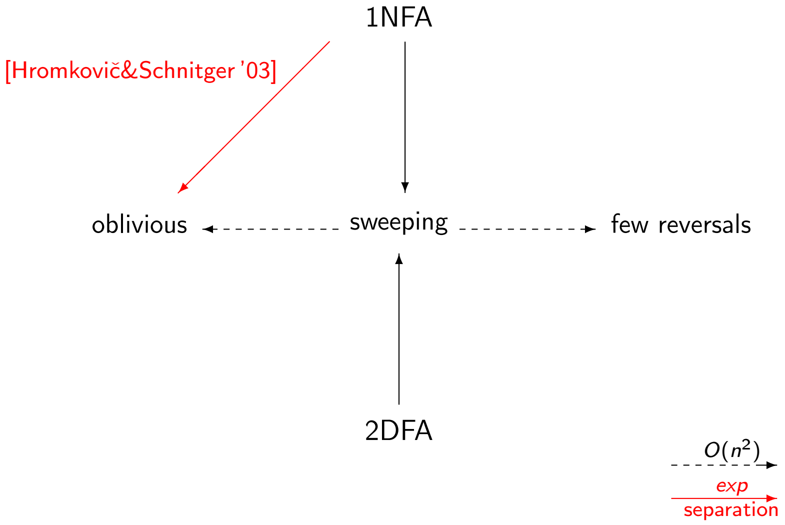




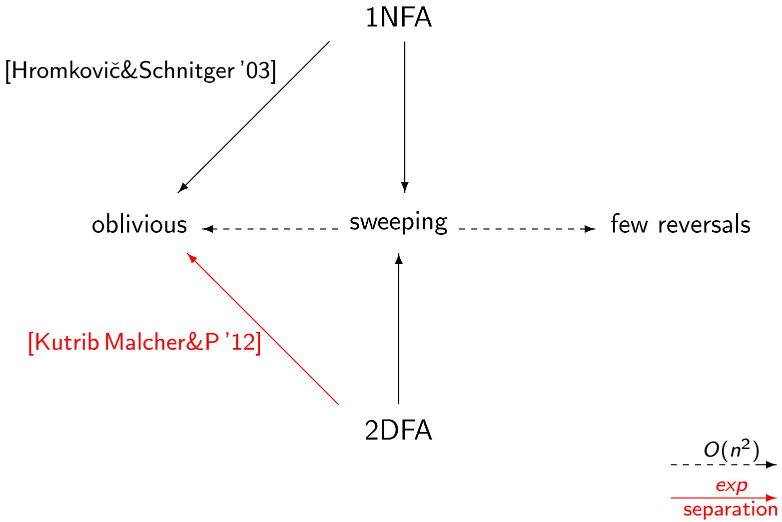
Restricted Models: Separations



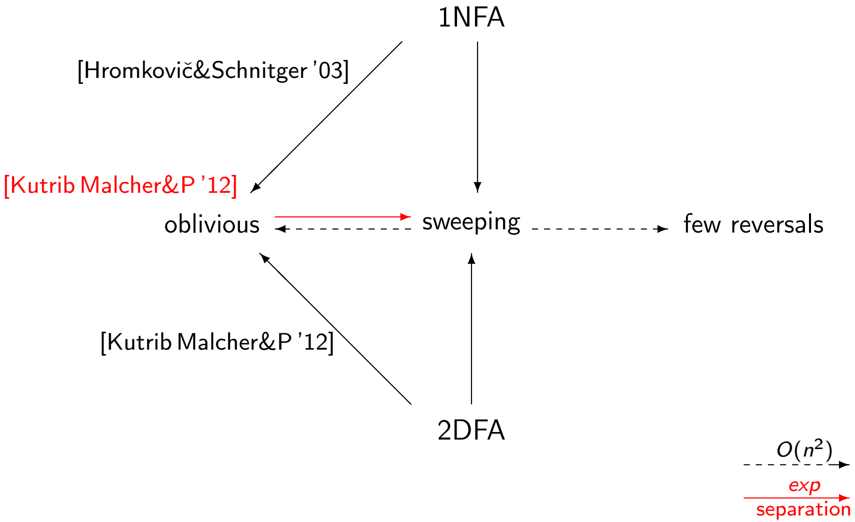
Restricted Models: Separations



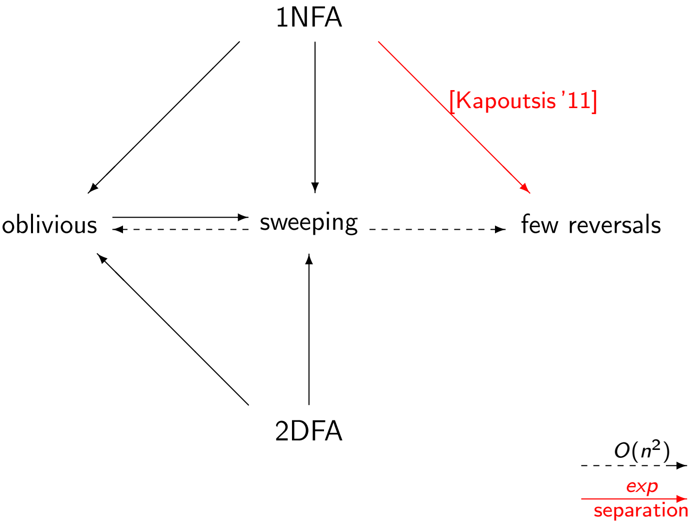
Restricted Models: Separations



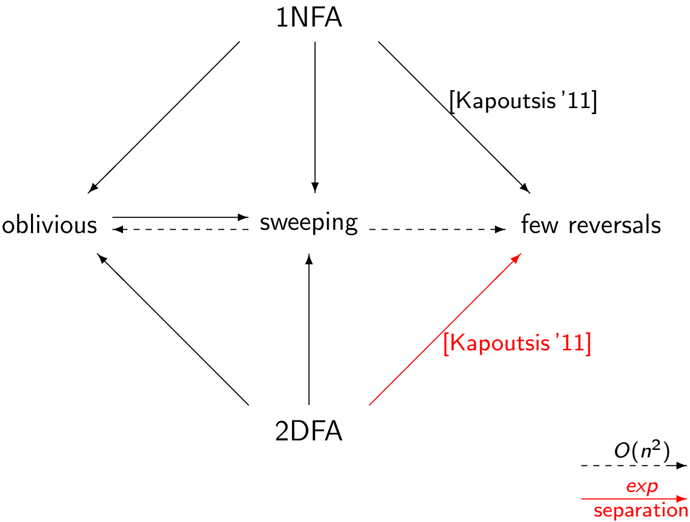
Restricted Models: Separations



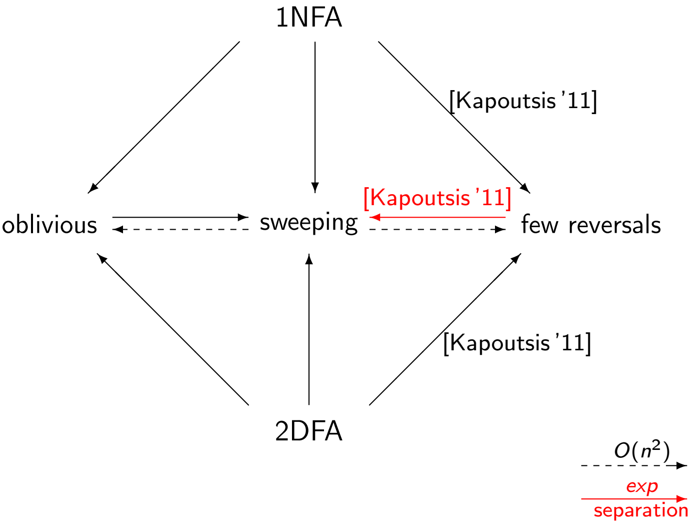
Restricted Models: Separations



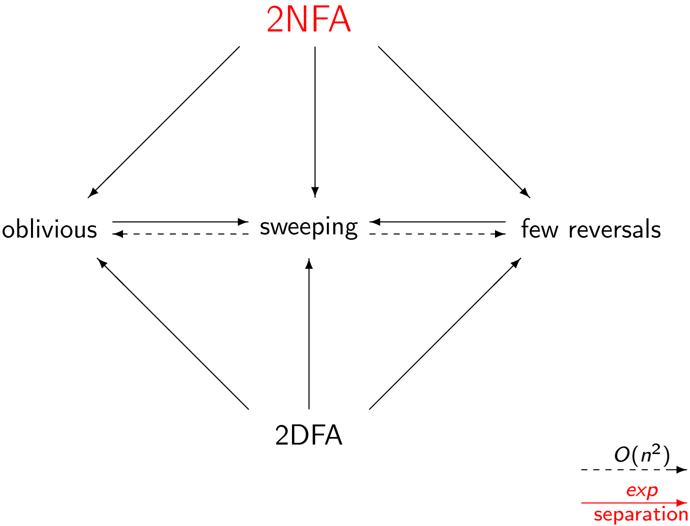
Restricted Models: Separations



Restricted Models: Separations

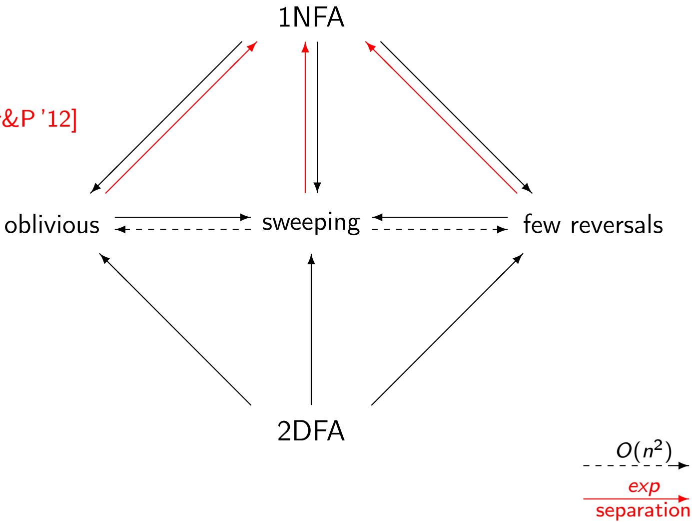


Restricted Models: Separations



Restricted Models: Separations

[Kutrib Malcher&P '12]



Sakoda&Sipser Question

Problem ([Sakoda&Sipser '78])

Do there exist polynomial simulations of

- *1NFAs by 2DFAs*
- *2NFAs by 2DFAs ?*

Another possible restriction:

The unary case $\#\Sigma = 1$

Optimal Simulation Between Unary Automata

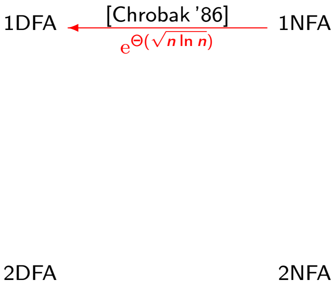
The costs of the optimal simulations between automata are different in the unary and in the general case

1DFA 1NFA

2DFA 2NFA

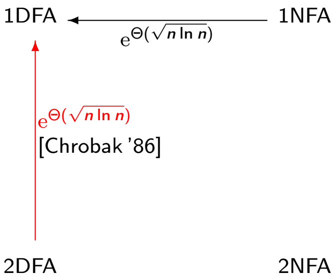
Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case



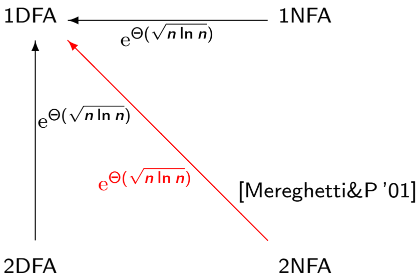
Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case



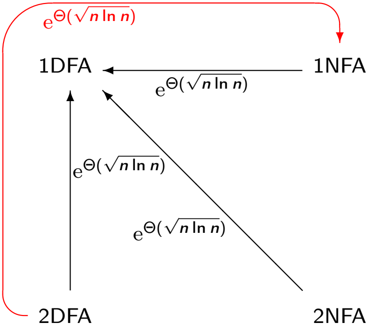
Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case



Optimal Simulation Between Unary Automata

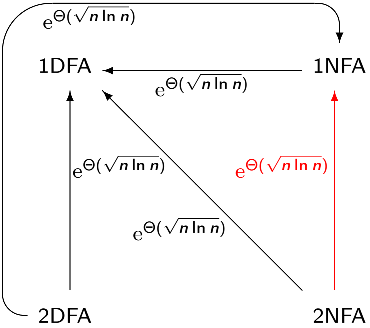
The costs of the optimal simulations between automata are different in the unary and in the general case



follows from 2DFA $\rightarrow$ 1DFA

Optimal Simulation Between Unary Automata

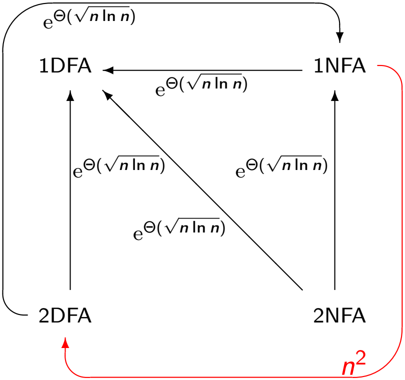
The costs of the optimal simulations between automata are different in the unary and in the general case



follows from 2NFA → 1DFA

Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case

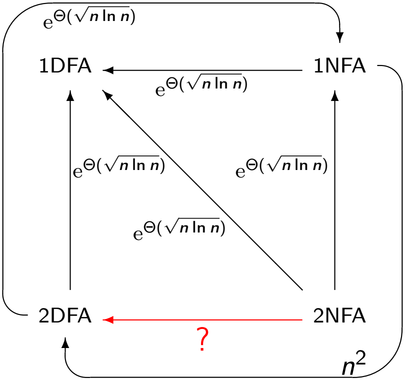


[Chrobak '86]

1NFA → 2DFA
In the unary case
this question is solved!
(polynomial conversion)

Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case

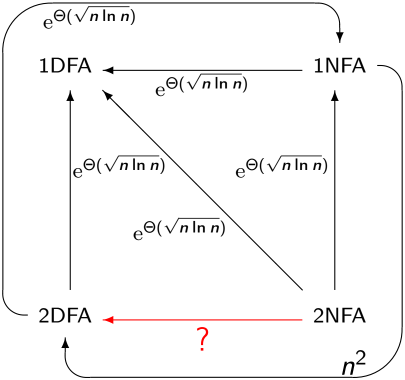


2NFA → 2DFA
Even in the unary case
this question is open!

- $e^{\Theta(\sqrt{n \ln n})}$ upper bound (from 2NFA → 1DFA)
- $\Omega(n^2)$ lower bound (from 1NFA → 2DFA)

Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case

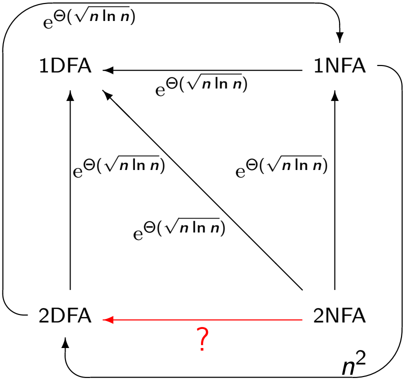


2NFA → 2DFA
Even in the unary case
this question is open!

- $e^{\Theta(\sqrt{n \ln n})}$ upper bound
  (from 2NFA → 1DFA)
- $\Omega(n^2)$ lower bound
  (from 1NFA → 2DFA)

Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case



2NFA → 2DFA
Even in the unary case
this question is open!

- $e^{\Theta(\sqrt{n \ln n})}$ upper bound
(from 2NFA → 1DFA)
- $\Omega(n^2)$ lower bound
(from 1NFA → 2DFA)

A better upper bound $e^{O(\ln^2 n)}$
has been proved!

A Normal Form for Unary 2NFAs

[Geffert Mereghetti&P '03]

Quasi Sweeping Automata (qsNFA):

- *nondeterministic choices* and
- *head reversals*

are *possible only* when the head is visiting the *endmarkers*

Theorem (Quasi Sweeping Simulation)

Each n-state unary 2NFA A can be transformed into a 2NFA M s.t.

- *M is quasi sweeping*
- *M has at most $N \leq 2n + 2$ states*
- *M and A are "almost equivalent"*
  (possible differences only for inputs of length $\leq 5n^2$)

A Normal Form for Unary 2NFAs

[Geffert Mereghetti&P '03]

Quasi Sweeping Automata (qsNFA):

- *nondeterministic choices* and
- *head reversals*

are *possible only* when the head is visiting the *endmarkers*

Theorem (Quasi Sweeping Simulation)

Each n-state unary 2NFA A can be transformed into a 2NFA M s.t.

- *M is quasi sweeping*
- *M has at most $N \leq 2n + 2$ states*
- *M and A are "almost equivalent"*
  (possible differences only for inputs of length $\leq 5n^2$)

From Unary qsNFAs to 2DFAs

[Geffert Mereghetti&P '03]

- M a fixed qsNFA with N states
- An input w is accepted iff there is an accepting computation visiting the left endmarker $\leq N$ times
- For $p, q \in Q$, $k \geq 1$, we define the predicate

  reachable$(p, q, k) \equiv \exists$*computation path on w which*
  - *starts in the state p on the left endmarker*
  - *ends in the state q on the left endmarker*
  - *visits the left endmarker $\leq k$ more times*
- Assuming acceptance on the left endmarker in state q_f:

  $w \in L(M)$ iff *reachable*(q_0, q_f, N) is true

From Unary qsNFAs to 2DFAs

[Geffert Mereghetti&P '03]

- M a fixed qsNFA with N states
- An input w is accepted iff there is an accepting computation visiting the left endmarker $\leq N$ times
- For $p, q \in Q$, $k \geq 1$, we define the predicate

  reachable$(p, q, k) \equiv \exists$*computation path on w which*
  - *starts in the state p on the left endmarker*
  - *ends in the state q on the left endmarker*
  - *visits the left endmarker $\leq k$ more times*
- Assuming acceptance on the left endmarker in state q_f:

  $w \in L(M)$ iff *reachable*(q_0, q_f, N) is true

From Unary qsNFAs to 2DFAs

[Geffert Mereghetti&P '03]

- M a fixed qsNFA with N states
- An input w is accepted iff there is an accepting computation visiting the left endmarker $\leq N$ times
- For $p, q \in Q$, $k \geq 1$, we define the predicate

  $$\text{reachable}(p, q, k) \equiv \exists \textit{computation path on } w \textit{ which}$$

  - *starts in the state p on the left endmarker*
  - *ends in the state q on the left endmarker*
  - *visits the left endmarker $\leq k$ more times*
- Assuming acceptance on the left endmarker in state q_f:

  $$w \in L(M) \text{ iff } \textit{reachable}(q_0, q_f, N) \text{ is true}$$

From Unary qsNFAs to 2DFAs
[Geffert Mereghetti&P '03]

- M a fixed qsNFA with N states
- An input w is accepted iff there is an accepting computation visiting the left endmarker $\leq N$ times
- For $p, q \in Q$, $k \geq 1$, we define the predicate

  reachable$(p, q, k) \equiv \exists$*computation path on w which*
  - *starts in the state p on the left endmarker*
  - *ends in the state q on the left endmarker*
  - *visits the left endmarker $\leq k$ more times*

- Assuming acceptance on the left endmarker in state q_f:

  $w \in L(M)$ iff *reachable*(q_0, q_f, N) is true

How to Evaluate *reachable*?

Divide–and–conquer technique

function *reachable*(p, q, k)
if $k = 1$ **then return** *reach1*(p, q) `//direct simulation`
else begin
  for each state $r \in Q$ **do**
    if *reachable*(p, r, $\lfloor k/2 \rfloor$) **and** *reachable*(r, q, $\lceil k/2 \rceil$)
      then return true `//recursion`
  return false
end

How to Evaluate *reachable*?

Divide–and–conquer technique

function *reachable*(p, q, k)
if $k = 1$ **then return** *reach1*(p, q) `//direct simulation`
else begin
  for each state $r \in Q$ **do**
    if *reachable*(p, r, $\lfloor k/2 \rfloor$) **and** *reachable*(r, q, $\lceil k/2 \rceil$)
      then return true `//recursion`
  return false
end

How to Evaluate *reachable*?

Divide–and–conquer technique

function *reachable*(p, q, k)
if $k = 1$ **then return** *reach1*(p, q) //direct simulation
else begin
  for each state $r \in Q$ **do**
    if *reachable*(p, r, $\lfloor k/2 \rfloor$) **and** *reachable*(r, q, $\lceil k/2 \rceil$)
      then return true //recursion
  return false
end

How to Evaluate *reachable*?

Divide–and–conquer technique

```
function reachable(p, q, k)
if k = 1 then return reach1(p, q)        //direct simulation
else begin
   for each state r ∈ Q do
      if reachable(p, r, ⌊k/2⌋) and reachable(r, q, ⌈k/2⌉)
         then return true                //recursion
   return false
end
```

How to Evaluate *reachable*?

Divide–and–conquer technique

function *reachable*(p, q, k)
if $k = 1$ **then return** *reach1*(p, q) `//direct simulation`
else begin
  for each state $r \in Q$ **do**
    if *reachable*$(p, r, \lfloor k/2 \rfloor)$ **and** *reachable*$(r, q, \lceil k/2 \rceil)$
      then return true `//recursion`
  return false
end

This strategy can be implemented by a 2DFA with $e^{O(\ln^2 N)}$ states
in order to compute *reachable*(q_0, q_f, N),
i.e., to decide if the input $w \in L(M)$

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		
M	almost equivalent qsNFA	$N \leq 2n+2$ states
$\Downarrow$		
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n + 2$ states
$\Downarrow$		
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n + 2$ states
$\Downarrow$		
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n+2$ states
$\Downarrow$		Subexponential Deterministic Simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n + 2$ states
$\Downarrow$		Subexponential Deterministic Simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n + 2$ states
$\Downarrow$		Subexponential Deterministic Simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n + 2$ states
$\Downarrow$		Subexponential Deterministic Simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

From Unary 2NFAs by 2DFAs

A	given unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n + 2$ states
$\Downarrow$		Subexponential Deterministic Simulation
B	2DFA equivalent to M	$e^{O(\ln^2 N)}$ states
$\Downarrow$		Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of B for longer inputs
C	2DFA equivalent to A	$e^{O(\ln^2 n)}$ states

Theorem ([Geffert Mereghetti&P '03])

Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states

Quasi Sweeping Simulation: Consequences

Using quasi sweeping simulation of unary 2NFAs several results have been discovered:

(i) Subexponential simulation of unary 2NFAs by 2DFAs
Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states [Geffert Mereghetti&P '03]

(ii) Polynomial complementation of unary 2NFAs
Inductive counting argument for qsNFAs [Geffert Mereghetti&P '07]

(iii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition $\mathrm{L} = \mathrm{NL}$ [Geffert&P '11]

(iv) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P '11]

Quasi Sweeping Simulation: Consequences

Using quasi sweeping simulation of unary 2NFAs several results have been discovered:

(i) Subexponential simulation of unary 2NFAs by 2DFAs
Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states [Geffert Mereghetti&P '03]

(ii) Polynomial complementation of unary 2NFAs
Inductive counting argument for qsNFAs
[Geffert Mereghetti&P '07]

(iii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition $\mathrm{L} = \mathrm{NL}$ [Geffert&P '11]

(iv) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P '11]

Quasi Sweeping Simulation: Consequences

Using quasi sweeping simulation of unary 2NFAs several results have been discovered:

(i) Subexponential simulation of unary 2NFAs by 2DFAs
Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states [Geffert Mereghetti&P '03]

(ii) Polynomial complementation of unary 2NFAs
Inductive counting argument for qsNFAs
[Geffert Mereghetti&P '07]

(iii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition $\mathrm{L} = \mathrm{NL}$ [Geffert&P '11]

(iv) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P '11]

Quasi Sweeping Simulation: Consequences

Using quasi sweeping simulation of unary 2NFAs several results have been discovered:

(i) Subexponential simulation of unary 2NFAs by 2DFAs
Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states [Geffert Mereghetti&P '03]

(ii) Polynomial complementation of unary 2NFAs
Inductive counting argument for qsNFAs [Geffert Mereghetti&P '07]

(iii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition $\mathsf{L} = \mathsf{NL}$ [Geffert&P '11]

(iv) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P '11]

Quasi Sweeping Simulation: Consequences

Using quasi sweeping simulation of unary 2NFAs several results have been discovered:

(i) Subexponential simulation of unary 2NFAs by 2DFAs
Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states [Geffert Mereghetti&P '03]

(ii) Polynomial complementation of unary 2NFAs
Inductive counting argument for qsNFAs
[Geffert Mereghetti&P '07]

(iii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition L = NL [Geffert&P '11]

(iv) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs
(unconditional) [Geffert&P '11]

Restricted 2NFAs

Outer Nondeterministic Automata (OFAs) [Guillon Geffert&P '12]:

- *nondeterministic choices*

are *possible only* when the head is visiting the *endmarkers*

Restricted 2NFAs

Outer Nondeterministic Automata (OFAs) [Guillon Geffert&P '12]:

- *nondeterministic choices*

are *possible only* when the head is visiting the *endmarkers*

Hence:

- No restrictions on the *input alphabet*
- No restrictions on *head reversals*
- *Deterministic transitions* on "real" input symbols

Outer Nondeterministic Automata (OFAs)

The results we obtained for the unary case
can be extended to 2OFAs: [Guillon Geffert&P '12]

(i) Subexponential simulation of 2OFAs by 2DFAs
(ii) Polynomial complementation of 2OFAs
(iii) Polynomial simulation of 2OFAs by 2DFAs
under the condition L = NL
(iv) Polynomial simulation of 2OFAs by unambiguous 2OFAs

While in the unary case all the proofs rely
on the *quasi sweeping simulation*,
for 2OFAs we do not have a similar tool!

Outer Nondeterministic Automata (OFAs)

The results we obtained for the unary case
can be extended to 2OFAs: [Guillon Geffert&P '12]

(i) Subexponential simulation of 2OFAs by 2DFAs

(ii) Polynomial complementation of 2OFAs

(iii) Polynomial simulation of 2OFAs by 2DFAs
under the condition L = NL

(iv) Polynomial simulation of 2OFAs by unambiguous 2OFAs

While in the unary case all the proofs rely
on the *quasi sweeping simulation*,
for 2OFAs we do not have a similar tool!

Outer Nondeterministic Automata (OFAs)

The results we obtained for the unary case can be extended to 2OFAs: [Guillon Geffert&P '12]

(i) Subexponential simulation of 2OFAs by 2DFAs

(ii) Polynomial complementation of 2OFAs

(iii) Polynomial simulation of 2OFAs by 2DFAs *under the condition* $\mathsf{L} = \mathsf{NL}$

(iv) Polynomial simulation of 2OFAs by unambiguous 2OFAs

While in the unary case all the proofs rely
on the *quasi sweeping simulation*,
for 2OFAs we do not have a similar tool!

Outer Nondeterministic Automata (OFAs)

Procedure *reach*(p, q)

- Checks the existence of a computation segment
  - from the left endmarker in the state p
  - to the left endmarker in the state q
  - not visiting the left endmarker in between
- Critical point: infinite loops
  - Modification of a technique for the complementation of 2DFAs [Geffert Mereghetti&P '07], which refines a construction for space bounded TM [Sipser '80]

Outer Nondeterministic Automata (OFAs)

Procedure *reach*(p, q)

- Checks the existence of a computation segment
  - from the left endmarker in the state p
  - to the left endmarker in the state q
  - not visiting the left endmarker in between
- Critical point: infinite loops
  - Modification of a technique for the complementation of 2DFAs [Geffert Mereghetti&P '07], which refines a construction for space bounded TM [Sipser '80]

Outer Nondeterministic Automata (OFAs)

Procedure *reach*(p, q)

- Checks the existence of a computation segment
  - from the left endmarker in the state p
  - to the left endmarker in the state q
  - not visiting the left endmarker in between
- Critical point: infinite loops
  - Modification of a technique for the complementation of 2DFAs [Geffert Mereghetti&P '07], which refines a construction for space bounded TM [Sipser '80]

Outer Nondeterministic Automata (OFAs)

Procedure *reach*(p, q)

- Checks the existence of a computation segment
  - from the left endmarker in the state p
  - to the left endmarker in the state q
  - not visiting the left endmarker in between
- Critical point: infinite loops
  - Modification of a technique for the complementation of 2DFAs [Geffert Mereghetti&P '07], which refines a construction for space bounded TM [Sipser '80]

Loops involving endmarkers are also possible

- They can be avoided by observing that for each accepting computation visiting one endmarkers more than $|Q|$ times there exists a shorter accepting computation

Sakoda&Sipser Question: Current Knowledge

- Upper bounds

	1NFA→2DFA	2NFA→2DFA
unary case and OFAs	$O(n^2)$ optimal	$e^{O(\ln^2 n)}$
general case	exponential	exponential

Unary case [Chrobak '86, Geffert Mereghetti&P '03]
OFAs [Guillon Geffert&P '12]

- Lower Bounds
In all the cases, the best known lower bound is $\Omega(n^2)$
[Chrobak '86]

Final Remarks

Speaking about...

...Finite automata

usually we mean

One-way finite automata

Final Remarks

Speaking about...

...Finite automata

usually we mean

One-way finite automata

...Turing machines

usually we mean

Two-way Turing machines

Final Remarks

Speaking about...

...Finite automata

usually we mean

One-way finite automata

...Turing machines

usually we mean

Two-way Turing machines

Why this difference?

Final Remarks

Speaking about...

...Finite automata

usually we mean

One-way finite automata

...Turing machines

usually we mean

Two-way Turing machines

Why this difference?

In both cases:

- *Computability* aspects
- *Complexity* aspects

Final Remarks

Speaking about...

...Finite automata

usually we mean

One-way finite automata

...Turing machines

usually we mean

Two-way Turing machines

Why this difference?

In both cases:

- *Computability* aspects
- *Complexity* aspects

Minicomplexity

- Complexity theory of two-way finite automata

[Kapoutsis, DCFS 2012]

Final Remarks

- The question of Sakoda and Sipser is very challenging
- In the investigation of restricted versions many interesting and not artificial models have been considered
- The results obtained under restrictions, even if not solving the full problem, are not trivial and, in many cases, very deep
- Connections with space and structural complexity
  - questions
  - techniques
- Connections with number theory (unary automata)

Final Remarks

- The question of Sakoda and Sipser is very challenging
- In the investigation of restricted versions many interesting and not artificial models have been considered
- The results obtained under restrictions,
  even if not solving the full problem,
  are not trivial and, in many cases, very deep
- Connections with space and structural complexity
  - questions
  - techniques
- Connections with number theory (unary automata)

Final Remarks

- The question of Sakoda and Sipser is very challenging
- In the investigation of restricted versions many interesting and not artificial models have been considered
- The results obtained under restrictions,
  even if not solving the full problem,
  are not trivial and, in many cases, very deep
- Connections with space and structural complexity
  - questions
  - techniques
- Connections with number theory (unary automata)

Final Remarks

- The question of Sakoda and Sipser is very challenging
- In the investigation of restricted versions many interesting and not artificial models have been considered
- The results obtained under restrictions,
  even if not solving the full problem,
  are not trivial and, in many cases, very deep
- Connections with space and structural complexity
  - questions
  - techniques
- Connections with number theory (unary automata)

Final Remarks

- The question of Sakoda and Sipser is very challenging
- In the investigation of restricted versions many interesting and not artificial models have been considered
- The results obtained under restrictions,
  even if not solving the full problem,
  are not trivial and, in many cases, very deep
- Connections with space and structural complexity
  - questions
  - techniques
- Connections with number theory (unary automata)

Thank you for your attention!